DOCUMENT RESUME

ED 336 073 IR 015 080

AUTHOR van Merrienboer, Jeroen J. G.; And Others
TITLE Training for Reflective Expertise: A Four-Component Instructional Design Model for Complex Cognitive Skills.
PUB DATE 91
NOTE 48p.; Paper presented at the Annual Conference of the American Educational Research Association (Chicago, IL, April 3-7, 1991).
PUB TYPE Information Analyses (070) -- Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Cognitive Processes; Educational Environment; Educational Strategies; Foreign Countries; Industrial Training; *Instructional Design; *Models; Postsecondary Education; Problem Solving; Programing; *Skill Development; Thinking Skills
IDENTIFIERS Interactive Teaching

ABSTRACT

This paper presents a four-component instructional design model for the training of complex cognitive skills. In the analysis phase, each skill is broken down into a set of recurrent skills that remain highly consistent over various problem situations, and a set of non-recurrent skills that require variable performance over different situations. Components 1 and 2 relate to analyses that are conducted to describe the recurrent skills in terms of specific rules or procedures and the non-recurrent skills in terms of heuristics or systematic problem approaches. Components 3 and 4 relate to the knowledge analyses that are conducted to describe the facts and concepts that are prerequisite to perform recurrent skills or the complex schemata that are helpful to perform non-recurrent skills. In the design phase, instructional tactics are selected for each of the four components. Components 1 and 2 relate to the design of practice and the conditions leading to rule automation or schema acquisition. Components 3 and 4 relate to the design of information presentation. Finally, the composition of the training strategy is governed by selected instructional tactics, which pose constraints on the design of an interactive learning environment. The basic prediction of this model is that its application leads to "reflective expertise" and increased transfer performance, i.e., the ability to solve new problems on the basis of domain-specific procedures and heuristics. Applications of the model that support this prediction are discussed for the training of fault management, computer programming, and statistical analysis skills. (74 references) (Author/DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

Paper Presented at the AERA Annual Meeting, Chicago, April 3-7, 1991.

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED336073

Training for Reflective Expertise: A Four-Component Instructional Design Model for Complex Cognitive Skills

Jeroen J. G. van Merriënboer
University of Twente, Dept. of Instructional Technology

Otto Jelsma
Fokker Aircraft BV, Logistic Training Division

Fred G. W. C. Paas
University of Twente, Dept. of Instructional Technology

Send your correspondence concerning this article to:

Jeroen J.G. van Merriënboer
University of Twente
Dept. of Instructional Technology
P.O. Box 217
7500 AE Enschede, The Netherlands

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Jeroen J.G.
van Merrienboer

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR015080

BEST COPY AVAILABLE

Contents

1. Introduction
2. The Four Component Instructional Design Model
   2.1. Analysis
      2.1.1. Identification of component skills
      2.1.2. Skills analyses
      2.1.3. Knowledge analyses
   2.2. Design
      2.2.1. The design of practice
      2.2.2. The design of information presentation
      2.2.3. Composing the training strategy
   2.3. Reflective Expertise and Transfer: Predictions
3. Applications of the Four Component Model
   3.1. Fault Management
   3.2. Computer Programming
   3.3. Statistical Analyzing
4. Discussion

Abstract

This paper presents a four-component instructional design model for the training of complex cognitive skills. In the analysis phase, the skill is decomposed in a set of recurrent skills, that remain highly consistent over various problem situations, and a set of non-recurrent skills, that require variable performance over different situations. The components 1 and 2 relate to the skills analyses that are conducted to describe the recurrent skills in terms of "strong methods" (specific rules and/or procedures) and the non-recurrent skills in terms of "weak methods" (heuristics and/or systematic problem approaches). The components 3 and 4 relate to the knowledge analyses that are conducted to describe the knowledge that is prerequisite to perform recurrent skills (facts and concepts) or that is helpful to perform non-recurrent skills (complex schemata, such as causal nets, plans and process models). In the design phase, instructional tactics are selected for each of the four components. The components 1' and 2' relate to the design of practice and pertain to the conditions under which practice leads to compilation of domain-specific rules (rule automation) or to inductive processing (schema acquisition). The components 3' and 4' relate to the design of information presentation and pertain to the presentation of the information that is prerequisite to perform the recurrent and non-recurrent skills. Finally, the composition of the training strategy is governed by the selected instructional tactics; they pose constraints to the design of an interactive learning environment and the information that is herein presented. The basic prediction of the model is that its application leads to "reflective expertise" and increased transfer performance, or, the ability to solve new problems on the basis of (a) domain-specific procedures that both solve familiar aspects of the problem and free up processing resources, thanks to which (b) heuristics may be applied to solve non-familiar aspects of the problem by the use schemata. Applications of the model that give support to this basic prediction are discussed for the training of fault management, computer programming and statistical analyzing skills.

1. Introduction

This paper presents the outline of an instructional design model for training complex cognitive skills or high-performance skills. The model is related to recent psychological theories on learning and information processing. In addition, three applications of the model in different domains are presented. For the purpose of this paper, we adopt Schneider's (1985) definition of high-performance skills. He defined this type of skills as having at least the following three characteristics. First, the trainee must expend considerable time and effort to acquire an acceptable mastery-level (i.e., usually more than 100 hours of training). Second, a substantial number of individuals fail to develop proficiency. Third, there will be qualitative differences in performance between novices and experts. Examples of complex cognitive skills are fault-management skills in process industry, computer programming skills, statistic analyzing skills, air traffic control skills, production and inventory management skills, and military air weapons control skills.

Whereas many instructional design models (see Reigeluth, 1983a) offer detailed guidelines for training one-dimensional skills (i.e., skills that aim at homogeneous sets of learning objectives) required to perform for instance recall tasks, classification tasks, procedural tasks and causal reasoning tasks, only a few models (e.g., Fabiani, Buckley, Gratton, Coles, & Donchin, 1989; Fisk & Gallini, 1989; Myers & Fisk, 1987; Schneider, 1985) provide guidelines for training multi-dimensional, complex cognitive skills that involve several component skills (i.e., skills that aim at integrated sets of various objectives). Furthermore, in so far as these models exist, a straightforward application of their guidelines is often quite difficult because they are usually formulated on a rather high level of abstraction. Actually, this is not surprising because there are still many intriguing questions regarding the underlying cognitive processes in learning complex cognitive skills. For instance, there is the vivid debate on the long-lasting problem concerning the relative importance of domain-independent versus domain-specific skills (Perkins & Salomon, 1989). And, related to this issue, the discussion regarding the trainability of domain-independent skills and the problem of how to obtain transfer of acquired skills and knowledge to a set of new task demands in different situations.

Complex cognitive skills are highly problem-solving intensive, so that the "transfer-of-training problem" is directly relevant to the problem of how to train those skills. In addition, the problems to-be-solved are frequently characterized by an ill-defined goal state (i.e., several acceptable solutions exist) and an ill-defined initial state (i.e., it is not clear which information is relevant to the solution of

the problem). In general, problem solving may be conceptualized as finding a solution for a problem by means of the transference of skills and knowledge that are acquired in former situations under different circumstances.

Today, the lack of detailed instructional design models for training complex cognitive skills is considered to be a serious omission in the area of instructional technology. This is particularly true because, in a society that is characterized by fast technological changes, many routine tasks are taken over by machines and new tasks that require more and more flexible problem solving behavior are rapidly emerging. Consequently, there is an increasing need for effective and efficient instructional guidelines for the training of complex cognitive skills.

The goal of training programs for this type of skills cannot be restricted to obtain a level of proficiency at which the skills can be performed quickly, fluently, effortlessly, and relatively automatically (rule-based behavior). Instead, in instructional settings for the training of high performance skills one should aim at *reflective expertise* (Olsen & Rasmussen, 1989), or the ability to perform familiar aspects of a task highly automatically, thanks to which processing resources become available that may be used to interpret new, unfamiliar aspects of the problem situation in terms of generalized knowledge about the task. This combination of rule-based and knowledge-based behavior implies transfer of acquired skills and knowledge (Thorndyke & Hayes Roth, 1979). In this paper, it will be argued that the nature of reflective expertise itself offers points of contact for the development of instructional design models for the training of complex cognitive skills.

The structure of the paper is as follows. In the second section, the outline of a four-component instructional design model (4C-model) and its predictions regarding reflective expertise and transfer will be presented. The 4C-model can either be seen as a generalization of Van Merriënboer's model (1990a, 1990b) for training computer programming skills or as a specification of the ADAPT design model (Apply Delayed Automation for Positive Transfer; Jelsma, Van Merriënboer, & Bijlstra, 1990; Pieters, Jelsma, & Van Merriënboer, 1987), which prescribes in general terms instructional strategies and tactics to reach transfer of learned skills and knowledge. In the third section, application of the 4C-model will be illustrated by examples of instruction for fault-management skills in process industry, computer programming skills, and statistical analyzing skills. Finally, a discussion of the model will be presented, and its implications for future research in both cognitive science and instructional science will be considered.

2. The Four-Component Instructional Design Model

Figure 1 presents the basic elements of the 4C-model. Application of the model requires a systematic approach. The process of building a training strategy for complex cognitive skills essentially passes through two general phases: Analysis and design. In the following sections, these two phases will be further elaborated. Then, the main predictions of the 4C-model regarding reflective expertise and transfer of training will be explained.

INSERT FIGURE 1 ABOUT HERE

2.1. Analysis

The main goal of the analysis phase is to give a detailed description of all component skills that make up the complex cognitive skill. First, the component skills are identified and classified as either recurrent skills, which remain consistent over various problem situations, or non-recurrent skills, which require a variable performance over different situations. Recurrent skills may be further classified as either procedural or non-procedural. Then, the component skills are described by means of skills analyses and knowledge analyses. In the following sections, the activities conducted in the analysis phase will be further elaborated. First, the identification of component skills will be discussed. Then, the skills analyses for recurrent skills (component 1) and non-recurrent skills (component 2) as well as the knowledge analyses for recurrent skills (component 3) and non-recurrent skills (component 4) will be described.

2.1.1. Identification of Component Skills

The first step to be taken is to decompose the skill into two sets of component skills. One set of component skills is formed by recurrent skills; essentially, these skills are performed in a similar way over various problem situations. The other set is formed by non-recurrent skills; performance of these skills may vary considerably over various problem situations. Recurrent skills, which are sometimes called consistent component skills or closed skills (Schneider & Fisk, 1982), are characterized by the fact that invariant rules or procedures, invariant components of processing, or invariant sequences of information-processing components are used in successful performance (Fisk & Gallini, 1989).

For instance, in supervising automatically controlled processes, the operator from time to time consistently executes several standard procedures to detect possible out-of-bounds situations (procedural recurrent skill). In addition, keyboard operation is highly recurrent because the operator always uses the same keyboard to perform his supervisory control task (non-procedural recurrent skill). In the field of computer programming, recurrent skills involve the use of the editor and the interpreter or compiler, the selection of basic language commands, and the application of syntactic rules of the language. And, in statistical analyzing, recurrent skills mainly involve the application of computational algorithms and the generation of applicable sequences of equations.

Non-recurrent skills, which are sometimes called variable component skills or open skills, are characterized by a variable performance over various problem situations. These skills have almost by definition a non-procedural character; the exact procedure for executing a particular activity varies each time the task is performed. Hence, it really is justified to refer to these skills as "transfer tasks" (or soft skills, Reigeluth & Merrill, 1984). For instance, after an operator has detected an out-of-bounds situation, its cause has to be diagnosed. It seems clear that the process of diagnosing system failures is highly non-recurrent because the failures, or combinations of failures, typically vary from one problem situation to another. In computer programming non-recurrent skills involve, for instance, decomposition of programming problems into increasingly easier-to-solve subproblems and the composition of a program by putting pieces of programming code together. Finally, in statistical analyzing, non-recurrent skills involve strategic decisions about the statistical techniques, concepts and principles to use and the coupling of the information that is present in the problem situation to the proper constants and variables.

2.1.2. Skills Analysis

After the two sets of component skills have been identified, an in-depth analysis of all involved component skills has to be conducted. According to the 4C-model, different task-analytical methods should be used for recurrent and non-recurrent skills. Component 1 of the model relates to the analysis of recurrent skills and component 2 relates to the analysis of non-recurrent skills.

2.1.2.1. Component 1: Analysis of recurrent skills. For procedural recurrent skills (e.g., starting up the computer and entering the programming environment, submitting a program to the compiler), a behavioral task analysis (Gropper, 1974) may be used if the skill mainly consists of motor operations; an information

processing analysis (Merrill, 1987) may be used if the skill contains relatively many mental operations. Both analyses employ some kind of hierarchical analysis, in which a particular procedure is broken down into increasingly more specific steps or actions. Then, the actions are arranged in the correct order of execution. The results of the analysis may be represented by means of a list of tasks and (sub-)subtasks (behavioral analysis) or in a flow-chart, pseudo-program or structured outline (information processing analysis).

For non-procedural recurrent skills it is often difficult or even impossible to specify a correct sequence of operations (e.g., handling the keyboard and selecting the correct basic language commands). Thus, other types of analyses are required. In essence, the rules have to be described that specify under which conditions particular actions apply. One possible approach is to conduct a factor-transfer analysis (Reigeluth & Merrill, 1984). In this type of analysis the factors that vary from situation to situation are identified and coupled to correct actions. The results are represented in so-called decision rules. Another approach is to model performance of the skill in an expert- or production system, which contains a set of condition-action pairs that specify what should be done under particular circumstances.

It is important to note that for the first component of the presented four-component model, recurrent skills (both procedural and non-procedural) are analyzed in terms of "strong methods". The specified procedures and/or decision rules are highly domain-specific and basically algorithmic in nature. They have great power because their application warrants that recurrent goals will be obtained. In a psychological way of speaking, recurrent skills are analyzed as if they were automatic processes (Schneider & Shiffrin, 1977; Shiffrin & Schneider, 1977). This is done for a good reason: in the design phase instructional tactics will be selected that directly foster automation of rules. By automating the recurrent component skills, processing resources may become available that are necessary to perform the non-recurrent component skills.

2.1.2.2. Component 2: Analysis of non-recurrent skills. One way to analyze non-recurrent skills is to specify, at a heuristic level, the actions and methods that ensure a Systematic Approach to Problem solving (SAP-analysis, Mettes, Pilot, & Roossink, 1981). Essentially, a sequence of actions has to be described that characterizes an optimal approach to the kind of problems to-be-solved. Whereas the results of a SAP-analysis can be presented in a flow-chart (in this case often called a SAP-chart) or structured diagram, it is fundamentally different from an information processing analysis. A SAP-analysis leads to the description of a systematic problem approach that is quite general and has a heuristic

nature; an information processing analysis leads to a description of a task that is highly specific and has an algorithmic nature. For instance, the results of a SAP-analysis for decomposition of programming problems may be as follows: read the problem carefully, mark the input- and output conditions, classify the problem, try to recognize the problem as one that is already known, identify the necessary objects, identify the necessary actions to apply to the objects, and so forth (see Tromp, 1989).

Another approach to analyze non-recurrent skills is to model the skill in a production system containing domain-independent rules that interpret knowledge to solve new problem situations. Thus, whereas recurrent skills are analyzed in terms of strong methods, non-recurrent skills are analyzed in terms of weak methods. The specified SAP's or rules are highly domain-general and heuristic in nature. Application of these rules does not automatically warrant the obtainance of current goals. Instead, power is exchanged for flexibility. As a result, different problem situations can be solved by the heuristic use of the same available knowledge.

In a psychological way of speaking, non-recurrent skills are analyzed as if they were controlled processes. This is not because it is impossible to describe non-recurrent skills as if they were automated. However, there are both theoretical and practical reasons for this choice. From a theoretical point of view, we aim at a description of the exit-behavior of the learners. As the non-recurrent component skills greatly vary over problem situations, it is often highly implausible that they will be automated by the learners. And from a practical point of view, it is often impossible to foresee all particular problem situations and their related procedures. For instance, thousands of combinations of faults may occur in particular production processes so that it becomes practically impossible to describe all related specific rules or procedures.

Finally, it should be clear that the efficiency of the application of SAP-charts or heuristics heavily depends on the availability of a rich declarative knowledge base. The better a learner's knowledge about a particular domain is organized, the more likely it is that the controlled, heuristic processes lead to a solution. In the 4C-model, non-recurrent skills are mainly analyzed in terms of the application of weak methods that make use of a rich declarative knowledge base, or, cognitive schemata (see component 4). Consequently, in the design phase, instructional tactics will be selected that directly foster the acquisition of such schemata.

2.1.3. Knowledge Analysis

The goal of the knowledge analysis is to describe the declarative knowledge that is either prerequisite or helpful to perform the complex cognitive skill. According to the 4C-model, different forms of knowledge analysis are needed for recurrent and non-recurrent component skills. Component 3 of the model relates to the analysis of knowledge that is prerequisite to the performance of recurrent skills; component 4 relates to the analysis of knowledge that may be helpful to perform non-recurrent skills.

2.1.3.1. Component 3: Analysis of prerequisite knowledge to perform recurrent skills. This component pertains to the analysis of knowledge that is minimally required to be able to correctly perform recurrent skills. With respect to this analysis of prerequisite knowledge, the distinction between procedural and non-procedural recurrent skills is not relevant. Given the kinds of analyses conducted for the first component, the prerequisite knowledge will mainly consist of concepts and facts. Essentially, the knowledge analysis that is to be conducted entails submitting each step in a flow-chart, pseudo-program or structured outline, or each proposition in a decision rule, to the question what the learner has to know in order to be able to perform that step or to apply that rule correctly. In fact, concept analysis (see Tennyson & Cocchiarella, 1986), is also a form of hierarchical analysis because it is repeated for all identified concepts until each of the concepts is one that is already mastered by the learner prior to the present instruction. In case of large amounts of facts or related concepts, it may be a good idea to structure the factual knowledge by means of a cluster analysis and to identify the relations between the concepts in taxonomies (kind-of relations), partonomies (part-of relations) or other hierarchies.

2.1.3.2. Component 4: Analysis of supportive knowledge to perform non-recurrent skills. Whereas the knowledge analysis for recurrent skills (component 3) is relatively easy and straightforward, the knowledge analysis for non-recurrent skills (component 4) is more difficult and elaborated. The knowledge analysis related to non-recurrent skills results in a description of knowledge that is supportive to perform this type of skills. It is postulated that the knowledge will often have the form of so-called schemata; cognitive structures that serve as abstractions to assign particular (sub)problems, on the basis of their characteristics, to particular categories that require particular plans or actions for their solution. Thanks to the presence of such schemata, learners are able to interpret unfamiliar situations in terms of their generalized knowledge. A process

of utmost importance to the use of schemata is analogy. This weak method may be seen as a mapping process in which the generalized declarative knowledge is mapped on the concrete situation in order to solve a particular problem (Anderson & Thompson, 1987).

For skills that involve reasoning, schemata that contain causal relations may be described. In a principle-transfer analysis (Reigeluth & Merrill, 1984), the causal relations that underlay the performance of the learner are identified. Complex causal nets can be described in AND/OR-graphs (Collins & Stevens, 1983) or other kinds of causal nets. These graphs enable the presentation of causal dependencies between larger amounts of factors. For instance, in the area of fault management in process industry AND/OR-graphs may describe the causal relations involved in the malfunctioning of a particular controller. The malfunction may be caused by corrosion OR high temperature, corrosion may be caused by a combination of water AND oxygen, high temperature may be caused by a failure of a temperature controller OR hot weather OR a human error, etcetera.

For skills that embody decisions, schemata may be described that involve (local) plans to reach particular goals. Essentially, those plans may be viewed as stereotyped, abstract solutions that are related to particular (sub)problems. In a goal-plan analysis (Schank & Abelson, 1977; Soloway, 1985), plans are described and coupled to a hierarchy of goals that must be achieved to solve the problem. For instance, a hierarchy of programming plans can be described in the field of computer programming. High-level programming plans (such as a general input-process-output plan) are related to general goals and may be applied to a very wide range of programming problems; medium-level plans (such as looping structures with an initialization above the loop) are related to less general goals, and low-level plans (such as the blueprint for a statement to print the value of a variable) are related to highly specific goals and they are applicable to increasingly smaller ranges of (sub)problems.

Finally, many skills involve qualitative reasoning about dynamic processes and require that the learner has an abstract notion of the system, machinery or process that must be manipulated. The term "mental model" is often used to refer to this kind of knowledge (Gentner & Stevens, 1983). Whereas some researchers view this knowledge as fundamentally different from procedural and declarative knowledge, it may also be argued that people have a set of declarative knowledge structures (i.e., schemata) for representing the form, function and relations between various devices and a set of procedures for reasoning about the interactions among these devices (e.g., Anderson, 1988). One way or the other, the analysis of mental models is in a relatively immature state compared to the

analysis of distinct skills and declarative knowledge. But nevertheless, if non-recurrent component skills involve much qualitative reasoning about dynamic processes it seems important to describe the system, machinery or process that must be manipulated as clearly as possible.

2.2. Design

After analyzing the complex cognitive skill and the knowledge that is prerequisite or supportive to its performance, an effective training strategy should be designed. Within any training strategy for teaching a complex skill, a distinction can be made between the design of practice (procedural instruction) and the presentation of information (declarative instruction; van Merriënboer & Krammer, 1987). In our view, procedural instruction is, or should be, the central part of each training strategy because it relates to the instructional design for actually performing the skill, that is, to the design of the problems or situations that the learner will be confronted with. Then, declarative instruction should scaffold the performance of the skill by the presentation of information that is relevant to its acquisition; but, if expertise increases, the value of declarative instruction decreases.

Within the presented four-component model, components 1' and 2' relate to the design of practice and components 3' and 4' relate to the design of information presentation. In the design phase, instructional tactics have to be selected that effectively train each of the four components. Only after the various tactics have been selected, the training strategy can be composed. In the following sections, the activities conducted in the design phase will be further elaborated. First, the instructional tactics that are related to the design of practice (components 1' and 2') and to the design of information presentation (components 3' and 4') will be discussed. Then, the composition of the training strategy out of the selected tactics will be described.

2.2.1. The Design of Practice

A very pervasive, but nevertheless important observation is that complex cognitive skills can only be learned by (mentally) doing them. Thus, in a training system designed to teach those skills the importance of formal instruction diminishes and the importance of practice increases. According to the four-component model, the effects of practice are twofold. First, as a result of practice, compilative processes (e.g., Anderson, 1983, 1987) produce domain-specific procedures that may eventually be directly applied to perform the skill without

the intercession of heuristic, weak methods. This process of rule automation is particularly important to the first component of our model: The training of recurrent component skills. Second, as a result of practice, inductive processes (e.g., Carbonell, 1984, 1986) may create new schemata or adjust existing schemata to make them more in tune with experience. This process of schema acquisition is particularly important to the second component of our model: The training of non-recurrent component skills. The instructional tactics that are related to the two effects of practice will be discussed in the following sections.

2.2.1.1. Component 1': Selecting tactics that promote compilative processing. Practice may result in the acquisition of domain-specific procedures. In the design phase, component 1' pertains to the design for practicing recurrent component skills; the instruction is designed to promote a rapid acquisition of domain-specific, automated rules. Anderson (1983, 1987) identified knowledge compilation as an important process in the creation of procedures that eventually may directly control behavior. It both includes the incorporation of newly acquired knowledge in task-specific procedures and the "chuncking" of procedures that consistently follow each other in performing the skill. Knowledge compilation produces a considerable speedup in performance and implies a reduction of processing load because newly acquired knowledge need no longer be retrieved from memory and held active to be interpreted by more general procedures. With regard to the design of practice, it should be noted that knowledge compilation is an elementary information process that is not subject to strategic control: It is mainly a function of the amount of consistent practice.

Instructional tactics for the design of training that promote rapid compilative processing usually invite learners to mechanically and consistently repeat performance and are often associated with repeated imitation and drill. The term drill originates from physical education and is still used (especially in military circles) for repetitive practice of some physical or mental operation that has to become automatic (e.g., rifle-drill, typing, or droning the multiplication tables). If the algorithmic procedures or specific rules that specify a correct performance are known (component 1), the most common tactic to be used is single-step or step-by-step instruction. For instance, one may teach multiplication by providing each step of the multiplication procedure to the learner, and having the learner to imitate each step until the procedure is completed. By repeating this process over and over the need for presentation of the steps disappears and the procedure will ultimately become automated.

For recurrent component skills that are difficult and demanding, part-task and part-whole training techniques form the most common solution to practicing

these skills. Under this approach, the whole procedure is decomposed into parts, and learners are extensively trained on each part separately before they move to practicing the whole procedure. The procedure may be broken down by segmentation (distinct temporal or spatial parts), fractionation (distinct functional parts), or simplification (simpler to more complex parts; Wightman & Lintern, 1985). For instance, Landa (1983) breaks down complex procedures by segmentation and suggests to practice the parts in a forward-chaining manner (the "snowball approach"); Scandura (1983) breaks down the procedure by simplification (simple to complex paths in the whole procedure) and suggests to practice the parts from simple to complex. To conclude, whereas many part-task approaches exist, they all aim at a smooth automation of the complete skill by practicing parts of it in a pre-specified sequence.

2.2.1.2. Component 2': Selecting tactics that promote inductive processing. Practice not only results in the acquisition of domain-specific procedures, but also in the modification of declarative knowledge (Proctor & Reeve, 1988). Component 2' relates to the design for practicing non-recurrent component skills; the instruction is designed to promote the (re-)organization or (re)-structuring of underlying knowledge in schemata, that may subsequently be used by analogy to solve novel problem solving situations. Schema induction seems an important process in the adjustment of existing schemata to make them more in tune with experience, or in the creation of new schemata. For example, a more generalized schema may be produced if a set of successful solutions is available for a class of related problems, so that a schema may be created that abstracts away from the details; a more specific schema may be produced if a set of failed solutions is available for a class of related problems, so that particular conditions may be added to the schema which restrict its range of use. After useful schemata have been developed, they may be used by analogies to generate behavior in new, unfamiliar problem situations. Obviously, this will often be the case if no task-specific, automated procedures are available. With regard to the design of practice, it should be noted that schema induction seems to be a process that is--at least to a certain degree--subject to strategic control, thus, it may require the conscious attention and mindful abstraction from the learner. In addition, it should be noted that schemata that are formed by inductive processes may, like all declarative knowledge, be compiled into domain-specific procedures. Obviously, this will only be the case if a schema is consistently used over problem situations. Then, the compiled procedures may eventually directly produce the effect of the use of the schema by analogy (or other weak methods) without making reference to declarative knowledge (Lewis & Anderson, 1985).

From the literature, many instructional tactics are known that may have a positive effect on inductive processing, and thus on schema acquisition during practice. Whereas the following list is far from complete, it gives a good impression of some important instructional tactics that may be used for the design of practice in such a way that it promotes inductive processing:

1. *Variability of practice.* One of the best-known instructional principles to encourage learners to develop schemata, and one that almost consistently has resulted in beneficial effects on transfer of training, is the application of variability of practice (see for examples, Cormier & Hagman, 1987; Singley & Anderson, 1989). According to this tactic, non-recurrent component skills must be practiced under conditions that require the performance of different variants of the task over problem situations.
2. *Contextual interference.* Closely related to the first tactic is the application of contextual interference (Lee & Magill, 1985; Shea & Zimny, 1983). Application of this tactic implies that non-recurrent component skills should be practiced under conditions of high interference. This may be realized by repeatedly presenting a restricted set of problems that each require different solutions in a random order. Whereas variability of practice is generally considered as the determinant factor for transfer of training, the results of studies on contextual interference suggest that it is not the variability per se, but the way it is structured across acquisition problems which determines the extent of transfer of training.
3. *Rule based instruction.* Whereas this term may be misleading, it is used by Fisk & Gallini (1989) to indicate the presentation of different types of problems with common structural features together with integrated schema's of procedural and declarative knowledge. Thus, single components of instruction consist of information that is organized in--noncognitive--schemata and different types of problems, which can be solved by the use of the presented schemata. In the study of Fisk & Gallini, the rule based instruction led to higher transfer of acquired skills than single-step instruction.
4. *Emphasis manipulation approach.* Gopher, Weil & Siegel (1989) studied an innovative practice strategy that manipulates the emphasis or priority of selected subcomponents of a task, or in terms of the 4C-model, aspects of the non-recurrent skill. During practice, subjects are provoked to pay full attention to important aspects of the skill one at the time, and only pay attention to all aspects of the skill after quite extensive practice. According to Gopher et al. (1989), the approach is suitable to teach subjects strategies and lead them to incorporate those strategies in long-term schema's.

5. *Hierarchical approach.* This approach to practice (Frederiksen & White, 1989) is, at least partly, based on theories of mental models. According to the approach, a set of "problem environments" must be devised to focus on particular component skills, concepts and strategies that are important to the whole skill. Practice in the problem environments takes place in an order that follows from the hierarchical relations among these components, and aims at the development of the trainee's mental model.
6. *Problems with non-specific goals.* In the field of mathematics, Owen and Sweller (1985) demonstrated that decreasing the use of means-ends-analysis by reducing goal specificity of presented problems resulted in higher schema acquisition and improved transfer. In kinematics and geometry, Sweller (1988; Sweller, Mawer, & Ward, 1983)) found evidence that problems that do not include specific goals facilitate inductive processing more than conventional problems, and lead to higher transfer of acquired skills.
7. *Completion strategy.* Van Merriënboer (1990a, 1990b) proposed a form a practice, in which not the solving of increasingly complex problem situations, but the completion of increasingly larger parts of incomplete solutions formed the kernel of the training. By presenting this kind of problems, students are forced to study the (incomplete) solutions carefully in order to solve the problem. This is believed to promote inductive processing and schema acquisition because students will abstract away from the details of the presented incomplete solutions.
8. *Annotated, worked examples.* Another powerful procedural tactic that may promote schema acquisition is the presentation of problems in direct combination with worked examples for structurally similar problems; the worked examples provide a blue-print to approach the problems to be solved. Anderson, Boyle, Corbett and Lewis (1986) suggest that schema induction is further facilitated if the critical features in these examples are clearly identified, that is, when the examples are annotated with information about what they are supposed to illustrate.

2.2.2. The Design of Information Presentation

Whereas an effective design of practice is the core element to each training strategy for teaching a complex cognitive skill, it should be obvious that the presentation of information is necessary to scaffold a correct performance of the skill during practice. In general, the instruction should not overwhelm learners in the initial stages of training with a large amount of information that seems all

relevant to later performance of the skill. In addition, skill development and knowledge construction should take place jointly and, moreover, the importance of the presentation of information should decrease as expertise increases (i.e., a scaffolded learning environment).

According to the four-component model, two aspects can be distinguished with respect to the presentation of information. First, information has to be presented that is directly conditional to a correct performance of the skill, and in particular, to a correct performance of recurrent component skills. In the analysis phase, this information has been identified as the third component. According to the 4C-model, it may be presented in such a way that it is restrictedly encoded and suitable for rapid proceduralization (i.e., embedded in rules or productions; Anderson, 1983, 1987), so that it promotes rule automation. Second, information has to be presented that is helpful to a correct performance of the skill, and in particular, to a correct performance of non-recurrent component skills, because it offers schematic knowledge that may be used by analogy or other controlled processes to solve new problem situations. In the analysis phase, this information has been identified as the fourth component. According to the 4C-model, this information must be presented in such a way that it is elaboratively encoded, that is, connected to already existing knowledge so that it promotes schema acquisition. The instructional design principles that are related to the presentation of information will be discussed in the following sections.

2.2.2.1. Component 3': Selecting tactics that promote restricted encoding. For the design phase, this component pertains to the presentation of information that is relevant to the acquisition and performance of recurrent component skills. As for the first component, the instruction is designed to promote the rapid acquisition of domain-specific, automated rules. For compilation to occur (i.e., embedding the information in domain-specific procedures), the necessary information must be active in working memory while the problem is being solved. This can be reached by making the information directly available during problem solving, or by retrieving the knowledge from long-term declarative memory. In the second case, the knowledge may be declaratively encoded in a relatively isolated manner: It is not critical that it is embedded into existing schemata so that during acquisition no particular reference has to be made to related knowledge. However, in either case the new information that is provided with the instruction should be encoded correctly. Anderson (1987) has pointed to the importance of correctness of declarative encodings for introducing a skill to a learner. He argues that any procedure compiled from an incorrect representation of knowledge may be severely detrimental to later performance of the skill.

The question what information to present to the learners has mainly been answered during the analysis of components 1 and 3. First, it relates to the procedures or specific rules that describe the correct performance of the recurrent component skill (component 1); second, it relates to the facts or concepts that form an integral part of these strong methods (component 3). Usually, a "didactical specification" is needed to transform this information into a form that is suitable to present to the students (Resnick, 1976). Essentially, this didactical specification pertains to the formulation of rules, procedural steps, concepts or facts in terms that are clearly understandable for the learners the instruction is designed for.

Instructional tactics for the presentation of information that promote restricted encoding may either aim at the prior availability of the information in declarative memory or its direct availability during problem solving. If prior availability is desirable, the learner may be encouraged to maintain the new information in an active state just by repeating it over and over, either out loud or mentally. This mere repetition of information, or, rehearsal, is sometimes associated with rote learning. There is substantial research evidence that rehearsal is an efficient way to maintain a relative small amount of information in an active state, allowing excellent short-term memory performance and subsequent compilation in domain-specific procedures during practice. On the other hand, often there will be no need for prior availability of the information in declarative memory, so that it can only be presented when it is needed, that is, parts of the information are provided when they are needed during practice.

According to the 4C-model, three key aspects to the presentation of information are *partitioning*, *demonstration* and *scaffolding*. Partitioning is particularly important for recurrent component skills because only the presentation of relatively small amounts of new information at the same time can prevent processing overload; in addition, this information should have to be held relatively short in working memory in order to prevent working memory failures, which may result in incorrect compilations and negatively affect performance of the skill. Thus, the necessary declarative instruction should be compacted into its bare essentials and gradually elaborated during the skill acquisition process. This pertains to the presentation of the specific rules or procedures ("how-to" instruction) as well as to the presentation of necessary concepts and facts. For instance, a support or help system may be used in a step-by-step paradigm as discussed in the previous section. During practice, the system provides each applicable procedural step or rule at the moment it must be applied; in addition, the facts and concepts that are used in this step or rule are simultaneously explained. If it is not possible to provide the necessary partitions of information

directly during practice, one should strive to present them in such a way that they are easily accessible during practice. In agreement with this point of view, Carroll, Smith-Kerker, Ford and Mazur-Rimetz (1986, 1988) showed that standard text-editor manuals can be made much more effective if they have the form of an easily accessible "minimal manual" that is shortened and focused just on the information that is necessary to perform the skill.

Another tactic for the presentation of information that is relevant to the performance of recurrent component skills relates to demonstration. In general, one should not only provide the rules, procedures or concepts that are relevant to performing the skill (so-called "expository generalities"; Merrill, 1983), but also demonstrations that illustrate the application of rules or--parts of--procedures as well as examples that typify concepts ("expository instances"). Apparently, no generalities exist for the plain facts that have to be presented. Finally, the last principle pertains to scaffolding, which means that the presentation of information becomes increasingly superfluous to the performance of recurrent component skills as the learners gain expertise (e.g., Charney & Reder, 1986). For instance, in a support or help system one may first systematically present supportive information during practice, then only provide this information on request of the learner, and finally provide no supportive information at all.

2.2.2.2. Component 4': Selecting tactics that promote elaborative encoding. This component relates to the presentation of information that is relevant to the acquisition and performance of non-recurrent component skills; as for the second component, the instruction is designed to promote the acquisition of schemata that may subsequently be used as analogies to solve new problem situations. For elaboration to occur (i.e., enrichment of newly presented information by existing knowledge), the information must be integrated and linked to existing knowledge structures in order to provide a knowledge base that is highly organized and suitable to be operated upon by weak methods. Elaborative processes connect schemata with the new information and they infer information from the schemata that was not presented in the instruction. The result of elaboration is an increase of interconnections in and between schemata, which is thought to facilitate retrievability and usability because multiple retrieval routes to particular information become available. What learners already know about a topic is used to help them to structure and understand new information about that topic, so that learners are encouraged not only to rehearse the new information but in addition are stimulated to relate it to previous knowledge and form either new or adjusted schemata.

The question what information to present to the learners has mainly been answered during the analysis of components 2 and 4. First, it relates to the heuristics (domain-general rules, rules-of-thumb) and SAP-charts that describe effective problem approaches for non-recurrent component skills (component 2); second, it relates to the causal nets, local plans and other schematic structures, such as process or machine models, that underlay a productive application of these weak methods (component 4). With regard to the teaching of SAP-charts or heuristics, most researchers agree that it is helpful to present this information, after a didactical specification, to the learners (e.g., Mettes et al., 1981; Schoenfeld, 1979). However, with regard to the underlying schemata there is a vivid discussion among researchers who favor particular forms of either expository learning or (guided) discovery (McDaniel & Schlager, 1990). For instance, in the field of computer programming one might aim at the acquisition of schemata by explicitly presenting programming plans to students (e.g., Soloway, 1985) or by confronting the students during practice with many worked examples, in the form of concrete programs, to facilitate the acquisition of these schemata by inductive processing.

According to the presented 4C-model, the discussion on discovery learning and expository learning is obscured by the fact that the components 2' and 4' are often not distinguished. On the one hand, instructional strategies that favor problem solving activities in some form of (guided) discovery learning might well lead to useful schemata, provided that tactics are applied that promote inductive processing during problem solving (component 2'). This point of view is present in the fields of "situated cognition" and "constructionism", which claim that learners construct meaning (i.e., acquire schemata) from interacting with real events, or, interactive environments that simulate real events. Indeed, this approach warrants that new information about a particular domain is fully integrated with already existing knowledge. On the other hand, there seems to be no strong reason whatsoever for *not* simultaneously providing the learner with pre-specified causal nets, local plans or models that may be helpful and offer guidance to solve the problems in that particular domain (component 4'). The point is, according to the authors, that these approaches need not be seen as distinct alternatives, but merely as two aspects of instruction that can, and often should, complement each other.

In contrast to tactics that promote restricted encoding, instructional tactics that promote elaborative encoding of presented schema-like information focus primarily on the prior availability of this information in declarative memory in such a way that it is highly integrated with already existing knowledge. In that way, the chance that the information can be helpful to the performance of non-

recurrent component skills is increased. Together with the intrinsic complexity of the information, this leads to implications for partitioning the information. The amount of information that must be presented at the same time can be considerable. For instance, SAP-charts, causal nets or process models (e.g., a concrete computer model, a model of a distillation process) can be quite substantial to describe. The information will usually be presented before units of practice, and instructional tactics for information presentation have to be used that promote elaboration of the presented material. Such tactics encourage the learner to relate the new information to existing knowledge, that is, they promote meaningful learning by facilitating the assimilation of the new information to schemata that already exist in memory. A wide variety of such tactics has been put forward (e.g., Annett & Sparrow, 1985; Brooks & Dansereau, 1987; Mayer & Greeno, 1972; Merrill, 1983; Reigeluth, 1983b). For instance, well-known tactics are the application of advance organizers, metaphors and mnemonic systems and the encouragement to paraphrase particular pieces of information.

With regard to demonstration and scaffolding, essentially the same principles apply as for non-recurrent skills. Thus, one should not only present heuristics, SAP-charts or schema-like units that are relevant to performing the skill, but also demonstrations that illustrate the application of the heuristics or SAP-charts as well as concrete examples that characterize, for instance, causal nets (particular predictions that can be made), plans (particular instantiations of plans) or models (particular states of a process). Given the high level of abstraction, as compared to the information that is relevant to the performance of recurrent skills, it may be difficult to develop adequate demonstrations. Modelling, that is, close observation of an expert or instructor who is performing the non-recurrent skill and explaining why he is doing what he is doing (either on-the-job or in an instructional setting), will often offer an effective approach to the demonstration of non-recurrent skills. Finally, it should be noted that systematically decreasing the amount of scaffolding is less easy than for recurrent skills. This is due to the fact that the presented information *may* help to solve the current problem but does not necessarily do so. In most cases, it seems advisable to present the information that seems helpful to perform particular skills before units of practice (and the need to do this decreases as expertise increases), but also to keep this information available for the learners during the whole training process. For instance, learners that are trained in a simulator-based instructional system study particular pieces of information before units of practice, but also have this information in some form available during the whole training process.

2.2.3. Composing the Training Strategy

The activities that are performed in the design phase provide two sets of selected instructional tactics. The first set consists of tactics for procedural instruction that prescribe methods to design the environment in which the complex cognitive skill will be practiced; it pertains to the practice design for both the recurrent and non-recurrent component skills that make up the complex cognitive skill. The second set consists of tactics for declarative instruction that prescribe methods to present the information that must scaffold the correct performance of the skill during practice; it pertains to the presentation of information that is prerequisite to perform recurrent component skills as well as to the presentation of information that may yield schemata that may be helpful to the performance of non-recurrent component skills.

Whereas the composition of the training strategy is--to a certain degree--a creative process, the sets of selected tactics will heavily constrain the problem space of instructional design. First, the selected tactics for the design of practice pose strict constraints to the interactive learning environment that is developed to practice the skill. Particular selected tactics prescribe methods that must be applied in the environment to secure compilation, or rule automation for recurrent component skills, while other selected tactics prescribe methods that must be applied to secure inductive processing and schema acquisition during the performance of non-recurrent component skills. Second, the selected tactics for the presentation of information pose constraints to the amount and form of the presented information, as well as the timing of its presentation within the learning environment. Particular selected tactics prescribe methods that must be applied to present and demonstrate the information that is prerequisite to a correct performance of recurrent skills, while other selected tactics prescribe methods that must be applied to promote elaboration of information that is presented to support the performance of non-recurrent component skills.

2.3. Reflective Expertise and Transfer: Predictions

According to the 4C-model, reflective expertise is best characterized as the ability to solve new problems on the basis of (a) domain-specific procedures that solve familiar aspects of a problem and that free up processing resources, thanks to which (b) heuristics may be applied to interpret the current situation in terms of a rich declarative knowledge base, organized in schemata, to solve the unfamiliar aspects of the problem. Thus, there are two aspects to reflective expertise: The first aspect is related to components 1 and 3 (rule automation) and

the second aspect is related to components 2 and 4 (schema acquisition). According to this line of reasoning, reflective expertise in performing a complex cognitive skill is clearly more than the sum of its automatic parts (see also, Olsen & Rasmussen, 1989). If unskilled performance were simply less automatic than skilled performance, unskilled performers would be able to do whatever skilled performers could do, only less automatically, and clearly this is wrong.

Jelsma et al. (1990) extensively discussed the implications of the 4C-model for transfer of training. In short, two components seem to be relevant to transfer of training: Procedural overlap and knowledge-based transfer. With regard to procedural overlap, it is assumed that transfer from a learned task to a transfer task will be positive to the extent that the underlying procedure sets overlap. Obviously, this component is particularly important to recurrent component skills within the whole complex cognitive skill. The identification of the common set of specific rules or procedures allows quantitative predictions for transfer of training (e.g., Polson & Kieras, 1984; Singley and Anderson, 1988). Furthermore, the concept of procedural overlap naturally implies that there is also a set of procedures that are specific to the learned tasks and the transfer tasks.

If the size of the specific set is large, that is, if the transfer task increasingly deviates from the original training task, transfer of training can no longer be attributed exclusively to procedure overlap. Then, knowledge-based transfer may explain transfer to the extent that declarative knowledge structures, or schemata, are available from other problem solving situations that may help to solve the non-familiar aspects of the current problem situation. Thus, for this component, schemata provide a basis for transfer between different uses of the same knowledge. Obviously, this component is particularly important to the non-recurrent skills that form part of the whole complex cognitive skill. In the 4C-model, analogy is seen as a particularly important process to provide knowledge-based transfer; it is conceptualized as the mapping process in which schemata are used to solve new problems. In fact, analogy draws comparisons between the current problem situation and possibly relevant knowledge from other more or less similar problem situations. If no relevant knowledge is available, analogy fails and other less powerful processes must be used to perform the transfer task. And in the worst case, it may happen that no solution can be found.

In agreement with the ADAPT design model, the 4C-model postulates that the importance of knowledge-based transfer increases if the transfer becomes further, that is, if the size of the overlapping set of procedures decreases and the specific set increases. Furthermore, it is assumed that the processes that are responsible for knowledge-based transfer (and in particular, analogy) will be more successful to the extent that the knowledge is better organized into schemata.

This declarative knowledge representation in schemata is considered to be a sound basis for analogy to operate upon because multiple retrieval routes to particular information are available which offer the possibility to derive relevant information more easily. In other words, the relative importance of schema acquisition increases if the transfer tasks become more different from the original training tasks.

This leads us to the basic predictions of the 4C-model. In contrast to most other models, the 4C-model explicitly takes the knowledge-based component for performing complex cognitive skills into account, by identifying SAP-charts and heuristics as well as schemata that may help to make an effective use of them (components 2 and 4). In addition, instructional tactics are selected that promote adequate schema acquisition by inductive processing and elaboration (components 2' and 4'). Given these facts, it is hypothesized that the model opens the way to attain reflective expertise and hence reach better transfer performance than traditionally developed instruction. Specifically, it is predicted that the superiority of instruction that is developed according to the 4C-model over traditional forms of instruction will become more apparent for performing transfer tasks that are more different from the training tasks. Illustrations of application of the model that give support to this basic prediction will be presented in the next section.

3. Applications of the Four Component Model

Up to the present, the 4C-model has been applied to design instruction in the fields of fault management in process industry, introductory computer programming, and statistical analyzing. In all three domains, experiments have been conducted to test the model's predictions regarding reflective expertise and transfer, namely, that the application of the model will yield instruction that leads to higher transfer performance than traditional instruction and that this superiority will increase as the transfer tasks differ more from the original training tasks. Illustrative applications of the model and their related experiments will be briefly discussed in the following sections.

3.1. Fault Management

Because of the ongoing automation in process industry, operators primarily monitor the behavior of automated systems and they are only actively involved with the system in cases of suboptimal production levels or system failures. The skill of coping with system failures is a typical complex cognitive skill that at least incorporates the phases detection, diagnosis, and compensation. Operators

have to notice that the system they are controlling is not acting in conformity with expectations (detection). After detection, the cause of the undesired system state has to be found (diagnosis). And finally, compensatory or corrective actions have to be taken in order to stabilize the system as quickly as possible (compensation).

The training of process operators for supervisory control and, in particular, for fault management tasks is a complicated matter. Operators must be trained to handle a countless number of combinations of system failures that can occur in the process to-be-controlled. It needs no explaining that it is implausible to cope with all possible system failures or combinations of failures in one training program. And, even if this could be realized, it would be impossible to exactly anticipate all kinds of system failures. Yet, it is precisely for the purpose of handling these unforeseen situations that human operators are employed. Clearly, for skilled operators, reflective expertise is a prerequisite to be able to adequately perform their complicated task.

In traditional operator training, formal instruction heavily emphasizes the theory of system functioning and of elementary physics and chemistry. Duncan (1981) refers to this as the "educational" approach. Actual skills training is usually provided on the job, with the drawback that most faults irregularly or even seldom occur so that the process of coping with these failures cannot be extensively practiced. In addition, available research evidence suggests that the emphasis on theoretical aspects of system functioning in traditional operator training is disproportionate to the actual value of such knowledge. Instead, it is suggested that the contents of instruction should be directly related to what the operator may be required to do in interaction with the system. That is, the training program should be directed to develop a set of fault management procedures that enable adequate control of the system.

Within the framework of the 4C-model an alternative training strategy was designed which is characterized by this aforementioned radical shift towards 'procedure-based' training. Such training has repeatedly been found to be more effective than the educational approach (e.g., Morris & Rouse, 1985; Shepherd, 1986; Shepherd, Marshall, Turner, & Duncan, 1977). Particular attention was given to the acquisition of reflective expertise and transfer of training, because the learned skills definitely have to transfer to new situations. Given the theoretically infinite number of possible system failures, the operators must be able to handle faults, or combinations of faults, that they have never encountered before. The especially developed training simulator PROCESS (Program for Research on Operator Control in an Experimental Simulated Setting; Jelsma & Bijlstra, 1990) was used to conduct an initial experiment.

As a first step, the complex cognitive skill of fault management in PROCESS was broken down into recurrent skills and non-recurrent skills. Recurrent skills involve, for instance, display monitoring or visual scanning and keyboard operation in performing the fault management task (detection, diagnosis, and compensation of system failures). Non-recurrent skills mainly pertain to the diagnosing phase of the fault management task. To be able to center the training around the procedures that had to be executed in order to be able to detect, diagnose, and compensate particular faults, the recurrent and non-recurrent skills were carefully analyzed. The skills analysis resulted in a large set of procedures for handling faults or combinations of faults. Actually, execution of these procedures required both recurrent skills (e.g., selection of a particular display) and non-recurrent skills (e.g., finding the cause of a particular system failure). Then, the knowledge required to perform both the recurrent and non-recurrent skills was determined. Finally, an exemplary sample of faults and their related procedures was selected.

In one of the conducted experiments, the selected faults were introduced in the simulation program and their related procedures were extensively practiced during several days. Using PROCESS, it was possible to predefine both the types of faults and the time at which they should occur in the simulated water-alcohol distillation process. In addition to PROCESS, a decision support system containing prerequisite knowledge on procedures, facts, and concepts was used in the experimental training program. The experimental and the control training strategies differed with respect to the level of contextual interference (Component 3', inductive processing) under which the handling of the selected faults was practiced.

INSERT FIGURE 2 ABOUT HERE

As shown in Figure 2, training under high contextual interference conditions proved to lead to higher transfer performance than training under low contextual interference conditions (Jelsma, 1989; Jelsma & Bijlstra, 1988). In addition, when the procedural overlap between the learned tasks and the transfer tasks decreased, the superiority of the high contextual interference condition over the low contextual interference condition increased. This finding is fully in agreement with the predictions of the 4C-model and supports the view that practicing under high contextual interference conditions facilitated schema acquisition by inductive processing. In general, the selection of instructional

tactics as suggested in the 4C-model proved to be useful to the design of an adequate training strategy for operator training. In fact, the data show that the application of the model contributed to the acquisition of reflective expertise, because the learners were better able to manage completely new, not previously encountered faults.

3.2. Computer Programming

In most conventional introductory programming courses, practice mainly consists of the generation of increasingly complex computer programs. In these traditional courses, learning outcomes and transfer performance are usually very low (for an overview, see Linn, 1985; Pea & Kurland, 1984). Within the framework of the 4C-model, an alternative form of training was designed to increase reflective expertise and transfer performance. As a first requirement, the training had to facilitate automation of those component skills involved in programming that are highly recurrent over problem situations, such as the use of the keyboard, the editor, and the interpreter or compiler, the selection of basic language commands, the application of the syntactic rules of the language, and so on. As a second, even more important requirement, the new form of practice had to facilitate schema acquisition for those component skills that heavily vary over problem situations, such as the decomposition of the programming problem and the subsequent composition of the program by putting programming language templates, that is, stereotyped patterns of programming code, together.

The resulting training strategy has been referred to as the "completion strategy" (Van Merriënboer & Krammer, 1990; Van Merriënboer & Paas, 1990). In this strategy, the students start with the running and hand-tracing of existing computer programs, then complete increasingly larger parts of well-structured, easily readable, but incomplete computer programs, and finally generate programs on their own. Thus, instead of the generation of increasingly complex computer programs, the completion of increasingly larger parts of incomplete computer programs forms the kernel of the strategy.

With regard to the design of practice, the completion strategy provides extensive training of component skills that are recurrent over problem situations, so that these skills may be readily automated. But moreover, it provides examples in the form of incomplete computer programs that may be annotated with information on what they are supposed to illustrate. These examples are directly available during practice and must be studied carefully in order to be able to correctly finish them; thus, they are expected to facilitate schema acquisition by inductive processing.

With regard to information presentation, the "minimal manual"-principle (Carroll et al., 1986, 1988) can be applied to present information on the use of the programming environment and the programming language that is prerequisite to perform non-recurrent skills. It is easily embedded in the completion strategy because all presented information can be directly illustrated by the programs that have to be completed, so that it is directly coupled to practice. Finally, measures were taken to promote elaboration of presented materials that are helpful to perform non-recurrent skills. For instance, three instructional tactics that can be applied pertain to the presentation of a concrete computer model (a model of how a computer works), general design schemata (SAP-charts of how to design a program), and programming language templates (plans that may be seen as the building blocks of programs).

In several experiments (Van Merriënboer, 1990a, 1990b; Van Merriënboer & De Croock, 1989), the effectiveness of the completion strategy was compared to conventional problem solving strategies. In a classroom study, high school students with an average age of 16 years and no prior programming experience received a 10-lesson introductory programming course. The course was given in a computer class for one period a week during 10 weeks, and its main goal was the acquisition of elementary programming skills for the computer language Comal-80. One group worked according to a traditional training strategy, based on workbooks that presented instructional texts (new information on the language and worked examples) and conventional programming problems for which new programs had to be generated. The other group worked according to a non-traditional training strategy, based on workbooks that presented identical instructional texts but so-called completion assignments. All completion assignments consisted of incomplete but readable programs, together with instructions to complete, extend or change the program to meet a given problem specification.

In a subsequent study on computer-based training (CBT; Van Merriënboer & de Croock, 1989), undergraduate social science students with an average age of 19 years and no prior programming experience worked on a CBT-program designed to teach elementary turtle-graphics programming techniques with Comal-80. One group received worked examples, which illustrated new information on features of the programming language and their syntactical details, and conventional problems that offered a problem description and some guidelines for designing and coding the program. The other group received completion assignments, that consisted of an incomplete program together with new information on programming language features that were illustrated by the

incomplete program, as well as instructions to complete, extend or change the program to meet the given problem specification.

INSERT FIGURE 3 ABOUT HERE

As may be seen in Figure 3, the completion strategy yielded higher or equal transfer performance than conventional training strategies in both studies. Low transfer tests measured the knowledge of the commands, syntax and standard language constructs of the programming language; high transfer tests measured proficiency in the design and construction of programs for *new* programming problems. For the low transfer tests, no significant differences between the two strategies were observed in the two studies. For the high transfer tests, the completion strategy was significantly superior to the conventional strategies in both studies. Thus, as predicted by the 4C-model, the superiority of the completion strategy over conventional strategies increased as the tests required more transfer of acquired skills. In further analyses, it appeared that the students who worked according to the completion strategy showed a superior use of programming language templates. As learned templates should be seen as a special kind of schemata, it seems highly plausible that the completion strategy indeed facilitated schema acquisition by inductive processing and so improved transfer performance due to reflective expertise.

3.3. Statistical Analyzing

Statistical analyzing is a complex cognitive skill, which involves much more than the straightforward application of statistical computational algorithms. Often, statistical problems are ill-defined; for instance, given a large amount of unorganized data a statistician may be confronted with the task to describe particular sets of data in terms of central tendencies (e.g., means, medians, modi), measures of dispersion, or other descriptive statistics. In such cases, statistical analyzing encompasses the separation of relevant and irrelevant information, strategic decisions on the statistical concepts, principles and techniques to be used, the generation of particular sequences of equations, the coupling of particular information to the proper constants and variables, and so forth.

Whereas the computations that are made for statistical analyzing are highly algorithmic, the problem situations are infinitely variable and each situation poses its own requirements to the concepts and techniques that are

optimal to use. For instance, learners may be well able to compute the mean, modus and median on a fixed array of numbers. But statistical analyzing also means that they must be able to decide on the statistics that are optimal to use and they must be able to compute these statistics in complex (real-life) situations in which the numbers are not pre-specified in a fixed array but must be searched for in often complex sets of data. Thus, training for reflective expertise is important to enable learners to make their statistical knowledge transferable. However, substantial evidence is available that in the domains of mathematics and physics conventional training strategies, which focus on solving goal-specific open problems, are not effective to reach such transfer (Cooper & Sweller, 1987; Larkin, McDermott, Simon, & Simon, 1980; Sweller, 1988, 1989; Sweller, Chandler, Tierney, & Cooper, 1990; Tarmizi & Sweller, 1988). Instead, the skills that are acquired in conventional training strategies are often limited to solving the kind of problems that are highly similar to the ones used during training.

Within the framework of the 4C-model, Paas (1991) developed two alternative training strategies for statistical analyzing: One strategy was based on the use of worked examples and the other on the use of completion assignments during practice. Initially, the recurrent and non-recurrent component skills involved in statistical analyzing (in particular, the description of data sets in central tendency measures) were identified. The recurrent skills mainly involve the application of computational algorithms and the generation of particular sequences of equations. Non-recurrent skills mainly relate to the selection of relevant information, the representation of the problem, and the strategic decisions on the statistics that are optimal to use. An information processing analysis on the recurrent component skills mainly revealed computational procedures; in addition, the concepts required to perform these recurrent component skills were determined. For the non-recurrent skills, the sequences of actions and methods that ensure effective statistical problem solving as well as the knowledge required to perform these non-recurrent skills were described.

The strategies emphasized, in order, the study of worked-out problem solutions and the completion of partly worked-out problem solutions. Thus, both strategies were characterized by the presentation of either complete or incomplete worked examples during practice. Such examples are expected to provide the possibility to extract general principles from the presented problem approaches, to focus attention on task-relevant aspects by minimizing the amount of cognitive resources that are required for extraneous activities, and to offer concrete schemata to map new solutions. In addition, they are expected to foster the organization of knowledge in memory and hence the accessibility of that knowledge during subsequent problem solving (Bassok & Holyoak, 1989). In short,

it was hypothesized that practice according to those strategies leads to more inductive processing and schema acquisition for the non-recurrent skills involved in statistical analyzing, resulting in higher transfer performance.

Paas (1991) compared the effectiveness of the two alternative training strategies with a traditional strategy, in which conventional problem solving formed the kernel of the training. The three training strategies were implemented in a computer-based training program that was designed to teach statistical analyzing skills to secondary school students with an age of 16-18 years. All students were offered identical formal instruction and the same set of statistical problems. For the conventional training strategy, all problems in the set were goal-specific open problems. For the worked strategy, the former two problems of each subset of three were problems together with worked-out problem approaches and solutions, while each third problem in the subset of three was an open problem. For the completion strategy, the former two problems in each subset of three were problems together with incomplete solutions that had to be completed, while each third problem was, again, an open problem.

INSERT FIGURE 4 ABOUT HERE

As shown in Figure 4, the training strategies that were based on worked examples and completion assignments yielded higher transfer performance than the conventional strategy which used goal-specific open problems. Low transfer tests measured the proficiency to solve statistical problems that were highly similar to the problems used during training and high transfer tests pertained to problems that significantly differed from those used during training. For both tests, training strategies based on worked examples or completion assignments were superior to the conventional strategies and this superiority increased if the tests required more transfer of acquired skills. These data support the predictions of the 4C-model and are in agreement with the results obtained in the fields of fault management and computer programming. To conclude this section, the 4C-model proved to be very useful to the conduct of instructional design for teaching complex cognitive skills in three different domains. And whereas the model is yet formulated on a general level, its fundamental concepts proved to lead to innovative training strategies that promote the development of reflective expertise and the acquisition of transferable skills.

4. Discussion

In this paper, the authors presented a four-component instructional design model for the training of complex cognitive skills. Two phases were distinguished: Analysis and design. In the analysis phase, the complex skill is decomposed in a set of recurrent skills, that remain highly consistent over various problem situations, and a set of non-recurrent skills, that require variable performance over different situations. Component 1 relates to the skills analyses that are conducted to describe the recurrent skills in terms of "strong methods", that is, specific rules or procedures that are basically algorithmic in nature. Component 2 relates to the skills analyses that are conducted to describe the non-recurrent skills in terms of "weak methods", that is, heuristics or systematic problem approaches. Component 3 relates to the knowledge analyses that are conducted to describe the knowledge that is prerequisite to perform recurrent skills; it encompasses the facts and concepts that form part of the identified rules or procedures. Finally, component 4 relates to the knowledge that is especially helpful to perform non-recurrent skills; this knowledge is believed to consists of complex schemata, such as causal nets, (local) plans and process models.

In the design phase, instructional tactics are selected for each of the four components. The components 1' and 2' relate to the design of practice, that is, they pertain to the conditions under which particular component skills must be practiced. For recurrent skills, effective practice should promote the compilation of domain-specific rules or rule automation, whereas for non-recurrent skills, effective practice must (also) promote inductive processing and schema acquisition. The components 3' and 4' relate to the design of information presentation, that is, they pertain to the content, form and timing of the information that is presented either before or during practice. The knowledge that is prerequisite to perform the recurrent skills (e.g., procedures, rules, concepts and facts) may be restrictedly encoded; in general, it is advisable to make this information directly available during practice. The knowledge that may be helpful to perform non-recurrent skills (e.g., heuristics, SAP-charts, causal nets, plans and models) are best elaboratively encoded; the information will often be presented before units of practice but this has to be done in such a way that it is optimally linked to already existing knowledge. Finally, the composition of the training strategy is governed by the selected instructional tactics; they pose strict constraints to the design of the interactive learning environment and the information that is herein presented.

The basic prediction of the 4C-model is that its application leads to "reflective expertise" and, consequently, to increased transfer performance.

Reflective expertise is best characterized as the ability to confront new problems on the basis of domain-specific procedures that solve familiar aspects of a problem; then, processing resources become available thanks to which heuristics may be applied to solve non-familiar aspects of the problem by the use of schemata. Thus, the first aspect of reflective expertise is related to the components 1 and 3 (rule automation) and the second aspect to the components 2 and 4 (schema acquisition). Applications that fitted the model in the fields of fault management in process industry, introductory computer programming, and statistical analyzing gave some support to the basic prediction: The instruction that was developed in accordance with the 4C-model yielded superior transfer performance and this superiority became more evident as the transfer tasks differed more from the training tasks. However, these studies clearly did not test the complete model but only particular elements of it. Specifically, they focused on the third component, or, the application of particular instructional tactics that promote schema acquisition by inductive processing.

Clearly, more research in instructional science is needed to test other elements of the 4C-model. Such research should both aim at the analysis and the design phase of the model. For the analysis phase, an important research question pertains to the decomposition of the skill in recurrent and non-recurrent component skills. For some complex cognitive skills, it is the authors' experience that it may prove to be very difficult to conduct an optimal decomposition of the skill, because there is a lack of good models for principled skill decomposition. Another research question is related to the analysis of complex knowledge structures, and in particular, the knowledge that underlies the ability to mentally simulate and reason about dynamic processes. Whereas such qualitative process models seem particularly important to the performance of complex cognitive skills that encompass trouble-shooting or fault management, the methods to analyze such models are in a relatively immature state compared to the methods to analyze skills, causal nets and plans.

In the design phase, a first concern for future research should be the extension, formulation and confirmation of instructional tactics that can possibly be applied in each of the four components. Tactics can be formulated in a goals-circumstances-method format (Van Merriënboer & Krammer, 1987). The goals are related to the desired outcomes of the instruction, the circumstances are factors that influence the effects of methods but cannot be manipulated, and the methods are manipulations to achieve different outcomes under different circumstances. Each instructional tactic should have at least one goal, often one or more circumstances to delimit its validity, and exactly one method. In the present article, the goals were only loosely described in terms of the cognitive processes

that are promoted by classes of instructional tactics that are related to one of the four components: Compilative processing, inductive processing, restricted encoding, and elaborative encoding. In addition, the circumstances under which those tactics should be applied were often not exactly described. Obviously, research should lead to a refinement of the goals and the circumstances in tactics to make an unambiguous selection of tactics for each of the four components possible.

To conclude, one may ask what the significance of the presented framework is for cognitive science. Given the 4C-model and the results that have been obtained by its application, the learning of heuristics and schematic, generalized declarative knowledge that may help to perform the non-recurrent components of complex cognitive skills (components 2 and 4) seems to be essential to the acquisition of such skills. This has direct implications for cognitive theories, and in particular for current theories of skill acquisition that describe the cognitive processes involved in learning complex cognitive skills. In particular, these theories focus on learning by doing (i.e., learning in conventional problem solving strategies) and rule automation (components 1 and 3) as the most essential processes in the acquisition of such skills. They seem to be incomplete with regard to their description of the acquisition of highly-organized declarative knowledge structures or schemata.

To sum up, it seems that instructional design may contribute to cognitive science, because the study of instructional design models such as the 4C-model and their resulting training strategies may provide evidence for the importance of particular cognitive processes that are neglected or underestimated by theories in cognitive science. Eventually, this may lead to the revision or extension of those cognitive theories. Whereas instructional design cannot lead to the "discovery" of new cognitive processes or a highly detailed description of those processes, the claim stands that studies in instructional design may provide evidence for the importance of particular cognitive processes, and thus lead to the integration of different perspectives in cognitive science. Such an integration of cognitive viewpoints is badly needed in order to be able to explain and predict human cognition and performance in newly developed instructional systems.

References

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192-210.

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems* (pp. 21-53). Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. (1986). *Cognitive modelling and intelligent tutoring* (Tech. Rep. No. ONR-86/1). Pittsburgh, PA: Carnegie-Mellon University, Dept. of Psychology.

Anderson, J. R., & Thompson, R. (1987). *Use of analogy in a production system architecture* (Tech. Rep.). Pittsburgh, PA: Carnegie Mellon University, Dept. of Psychology.

Annett, J., & Sparrow, J. (1985). Transfer of training: A review of research and practical implications. *Programmed Learning and Educational Technology, 22,* 116-124.

Bassok, M., & Holyoak, K. J. (1989). Interdomain transfer between isomorphic topics on algebra and physics. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 15,* 153-166.

Brooks, L. W., & Dansereau, D. F. (1987). Transfer of information: An instructional perspective. In S. M. Cormier & J. D. Hagman (Eds.), *Transfer of learning: Contemporary research and applications* (pp. 121-150). San Diego, CA: Academic Press.

Carbonell, J. G. (1984). Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michaelsky, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1, pp. 137-161). Berlin, Germany: Springer-Verlag.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michaelsky, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2, pp. 371-392). Los Altos, CA: Morgan Kaufman Publishers.

Carroll, J. M., Smith-Kerker, P. L., Ford, J. R., & Mazur-Rimetz, S. A. (1986). *The minimal manual* (IBM Research Rep. 11637). Yorktown Heights, NY: IBM Watson Research Center.

Carroll, J. M., Smith-Kerker, P. L., Ford, J. R., & Mazur-Rimetz, S. A. (1988). The minimal manual. *Human-Computer Interaction, 3,* 123-153.

Charney, D. H., & Reder, L. M. (1986). *Initial skill learning: An analysis of how elaborations facilitate the three components* (Tech. Rep.). Pittsburgh, PA: Carnegie-Mellon University, Dept. of Psychology.

Collins, A., & Stevens, A. L. (1983). A cognitive theory of inquiry teaching. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models* (pp. 247-278). Hillsdale, NJ: Lawrence Erlbaum.

Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology, 79,* 347-362.

Cormier, S. M., & Hagman, J. D. (Eds.). (1987). *Transfer of learning: Contemporary research and applications.* San Diego, CA: Academic Press.

Duncan, K. D. (1981). Training for fault diagnosis in industrial process plants. In J. Rasmussen & W. B. Rouse (Eds.), *Human detection and diagnosis of system failures* (pp. 553-573). New York, NY: Plenum Press.

Fabiani, M., Buckley, J., Gratton, G., Coles, M. G. H., & Donchin, E. (1989). The training of complex task performance. *Acta Psychologica, 71,* 259-299.

Fisk, A. D., & Gallini, J. K. (1989). Training consistent components of tasks: Developing an instructional system based on automatic/controlled processing principles. *Human Factors, 31,* 453-463.

Frederiksen, J. R., & White, B. Y. (1989). An approach to training based upon principled task composition. *Acta Psychologica, 71,* 89-146.

Gentner, D., & Stevens, A. L. (1983). *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum.

Gopher, D., Weil, M., & Siegel, D. (1989). Practice under changing priorities: An approach to the training of complex skills. *Acta Psychologica, 71,* 147-177.

Gropper, G. L. (1974). *Instructional Strategies.* Englewood Cliffs, NJ: Educational Technology Publications.

Jelsma, O. (1989). *Instructional Control of Transfer.* Enschede, The Netherlands: Bijlstra & Van Merriënboer.

Jelsma, O., & Bijlstra, J. P. (1988). Training for transfer in learning to detect, diagnose, and compensate system failures. *Proceedings of the Seventh European Annual Conference on Human Decision Making and Manual Control* (pp. 256-262). France, Paris.

Jelsma, O., & Bijlstra, J. P. (1990). PROCESS: Program for Research on Operator Control in an Experimental Simulated Setting. *IEEE Transactions on Systems, Man, and Cybernetics, 20*, 1221-1228.

Jelsma, O., van Merriënboer, J. J. G., & Bijlstra, J. P. (1990). The ADAPT design model: Towards instructional control of transfer. *Instructional Science, 19*, 89-120.

Landa, L. N. (1983). The algo-heuristic theory of instruction. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models* (pp. 163-211). Hillsdale, NJ: Lawrence Erlbaum.

Larkin, J., McDermott, J., Simon, D., & Simon, H. (1980). Models of competence in solving physics problems. *Cognitive Science, 4*, 317-348.

Lee, T. D., & Magill, R. A. (1985). Can forgetting facilitate skill acquisition? In D. Goodman, R. B. Wilberg, & I. M. Franks (Eds.), *Differing perspectives in motor learning, memory and control* (pp. 3-22). Amsterdam, The Netherlands: Elsevier Science Publishers.

Lewis, M. W., & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology, 17*, 26-65.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher, 14*(5), 14-29.

Mayer, R. E., & Greeno, J. G. (1972). Structural differences between learning outcomes produced by different instructional methods. *Journal of Educational Psychology, 63*, 165-173.

McDaniel, M. A., & Schlager, M. S. (1990). Discovery learning and transfer of problem-solving skill. *Cognition and Instruction, 7*, 129-159.

Merrill, M. D. (1983). Component display theory. In C. M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 278-333). Hillsdale, NJ: Lawrence Erlbaum.

Merrill, P. (1987). Job and task analysis. In R. M. Gagné (Ed.), *Instructional Technology: Foundations* (pp. 141-173). Hillsdale, NJ: Lawrence Erlbaum.

Mettes, C. T. W., Pilot, A., & Roossink, H. J. (1981). Linking factual knowledge and procedural knowledge in solving science problems: A case study in a thermodynamics course. *Instructional Science, 10*, 333-361.

Morris, N. M., & Rouse, W. B. (1985). The effects of type of knowledge upon human problem solving in a process control task. *IEEE Transactions on Systems, Man, and Cybernetics, 15*, 698-707.

Myers, G. L., & Fisk, A. D. (1987). Training consistent task components: Application of automatic and controlled processing theory to industrial task training. *Human Factors, 29*, 255-268.

Olsen, S. E., & Rasmussen, J. (1989). *The reflective expert and the prenovice: Notes on skill-, rule, and knowledge-based performance in the setting of instruction and training* (Tech. Rep.). Roskilde, Denmark: Risö National Laboratory.

Owen, E., & Sweller, J. (1985). What do students learn while solving mathematics problems? *Journal of Educational Psychology, 77*, 272-284.

Paas, F. G. W. C. (1991). *Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive load approach* (Tech. Rep. IST-MEMO-91-02). Enschede, The Netherlands: University of Twente.

Pea, R. D., & Kurland, M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*, 131-168.

Perkins, D. N., & Salomon, G. (1989). Are cognitive skills context-bound? *Educational Researcher, 18*, 16-25.

Pieters, J. M., Jelsma, O., & Van Merriënboer, J. J. G. (1987, September). *Skill acquisition: ADAPT instructional time to desired level of transfer.* Paper presented at the Second European Conference for Research on Learning and Instruction (EARLI), Tübingen, Germany.

Polson, P. G., & Kieras, D. E. (1984). A formal description of users' knowledge of how to operate a device and user complexity. *Behavior Research, Methods, Instruments, & Computers, 16*, 249-255.

Proctor, R. W., & Reeve, T. G. (1988). The acquisition of task-specific productions and modification of declarative representations in spatial-precueing tasks. *Journal of Experimental Psychology: General, 117*, 182-196.

Reigeluth, C. M. (Ed.) (1983a). *Instructional Design Theories and Models: An overview of their current status*. Hillsdale, NJ: Lawrence Erlbaum.

Reigeluth, C. M. (1983b). Meaningfulness and instruction relating what is being learned to what a student knows. *Instructional Science, 12*, 197-218.

Reigeluth, C. M., & Merrill, M. D. (1984). *Extended Task Analysis Procedures (ETAP): User's Manual*. Lanham, MD: University Press of America.

Resnick, L. B. (1976). Task analysis in instructional design: Some cases from mathematics. In D. Klahr (Ed.), *Cognition and Instruction* (pp. 51-80). Hillsdale, NJ: Lawrence Erlbaum.

Scandura, J. M. (1983). Instructional strategies based on the structural learning theory. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models* (pp. 213-246). Hillsdale, NJ: Lawrence Erlbaum.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding*. Hillsdale, NJ: Lawrence Erlbaum.

Schneider, W. (1985). Training high-performance skills: Fallacies and guidelines. *Human Factors, 27*, 285-300.

Schneider, W., & Fisk, A. D. (1982). Degree of consistent training: Improvements in search performance and automatic process development. *Perceptions & Psychophysics, 31*, 160-168.

Schneider, W., & Shiffrin, R. M. (1977). Controlled and automatic human information processing: I. Detection, search, and attention. *Psychological Review, 84*, 1-66.

Schoenfeld, A. H. (1979). Can heuristics be taught? In J. Lochhead & J. Clement (Eds.), *Cognitive Process Instruction* (pp. 315-338). Philadelphia, VA: Franklin Institute Press.

Shea, J. B., & Zimny, S. T. (1983). Context effects in memory and learning movement information. In R. A. Magill (Ed.), *Memory and control of action* (pp. 345-366). Amsterdam, The Netherlands: Elsevier North-Holland.

Shepherd, A. (1986). Issues in the training of process operators. *International Journal of Industrial Ergonomics, 1*, 49-64.

Shepherd, A., Marshall, E. C., Turner, A., & Duncan, K. D. (1977). Diagnosis of plant failures from a control panel: A comparison of three training methods. *Ergonomics, 20*, 347-361.

Shiffrin, R. M., & Schneider, W. (1977). Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review, 84*, 127-190.

Singley, M. K., & Anderson, J. R. (1988). A keystroke analysis of learning and transfer in text editing. *Human-Computer Interaction, 3*, 223-274.

Singley, M. K., & Anderson, J. R. (Eds.) (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.

Soloway, E. (1985). From problems to programs via plans: The content and structure of knowledge for introductory LISP programming. *Journal of Educational Computing Research, 1*, 157-172.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*, 257-285.

Sweller, J. (1989). Cognitive technology: Some procedures for facilitating learning and problem solving in mathematics and science. *Journal of Educational Psychology, 4*, 457-466.

Sweller, J., Chandler, P., Tierney, P., & Cooper, M. (1990). Cognitive load as a factor in the structuring of technical material. *Journal of Experimental Psychology: General, 119*, 176-192.

Sweller, J., Mawer, R., & Ward, M. (1983). Development of expertise in mathematical problem solving. *Journal of Experimental Psychology: General, 112*, 634-656.

Tarmizi, R. A., & Sweller, J. (1988). Guidance during mathematical problem solving. *Journal of Educational Psychology, 80*, 424-436.

Tennyson, R. D., & Cocchiarella, M. J. (1986). An empirically based instructional design theory for teaching concepts. *Review of educational Research, 56*, 40-71.

Thorndyke, P. W., & Hayes-Roth, B. (1979). The use of schemata in the acquisition and transfer of knowledge. *Cognitive Psychology, 11*, 82-106.

Tromp, Th. J.M. (1989). *The Acquisition of Expertise in Computer Programming Skill*. Amsterdam, The Netherlands: Thesis Publishers.

Van Merriënboer, J. J. G. (1990a). Strategies for programming instruction in high school: Program completion vs. program generation. *Journal of Educational Computing Research, 6*, 265-287.

Van Merriënboer, J. J. G. (1990b). *Teaching introductory computer programming: A perspective from instructional technology*. Enschede, The Netherlands: Bijlstra & Van Merriënboer.

Van Merriënboer, J. J. G., & De Croock, M. B. M. (1989, september). *Strategies for computer-based programming instruction: Program completion vs. program generation*. Paper presented on the Third European Conference for Research on Learning and Instruction (EARLI), Madrid, Spain.

Van Merriënboer, J. J. G., & Krammer, H. P. M. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science, 16*, 251-285.

Van Merriënboer, J. J. G., & Krammer, H. P. M. (1990). The "completion strategy" in programming instruction: Theoretical and empirical support. In S. Dijkstra, B. H. M. Van Hout-Wolters, & P. C. Van der Sijde (Eds.), *Research on instruction* (pp. 45-61). Englewood Cliffs, NJ: Educational Technology Publications.

Van Merriënboer, J. J. G., & Paas, F. G. W. C. (1989). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior, 6*, 273-289.

Wightman, D. C., & Lintern, G. (1985). Part-task training for tracking and manual control. *Human Factors, 27*, 267-284.

Figure Captions

Figure 1. An outline of the four-component instructional design model.

Figure 2. Time scores (A) and error scores (B) for low and high transfer tests after training fault management skills under conditions of high contextual interference and low contextual interference (conventional).

Figure 3. Percentages of correct operations for low and high transfer tests after training computer programming skills under the completion strategy and a conventional problem solving strategy in classroom teaching (A) and computer-based training (B).

Figure 4. Percentages of correct operations for low and high transfer tests after training statistical analyzing skills under the worked-examples strategy vs. a conventional problem solving strategy (A) and the completion strategy vs. the conventional strategy (B).

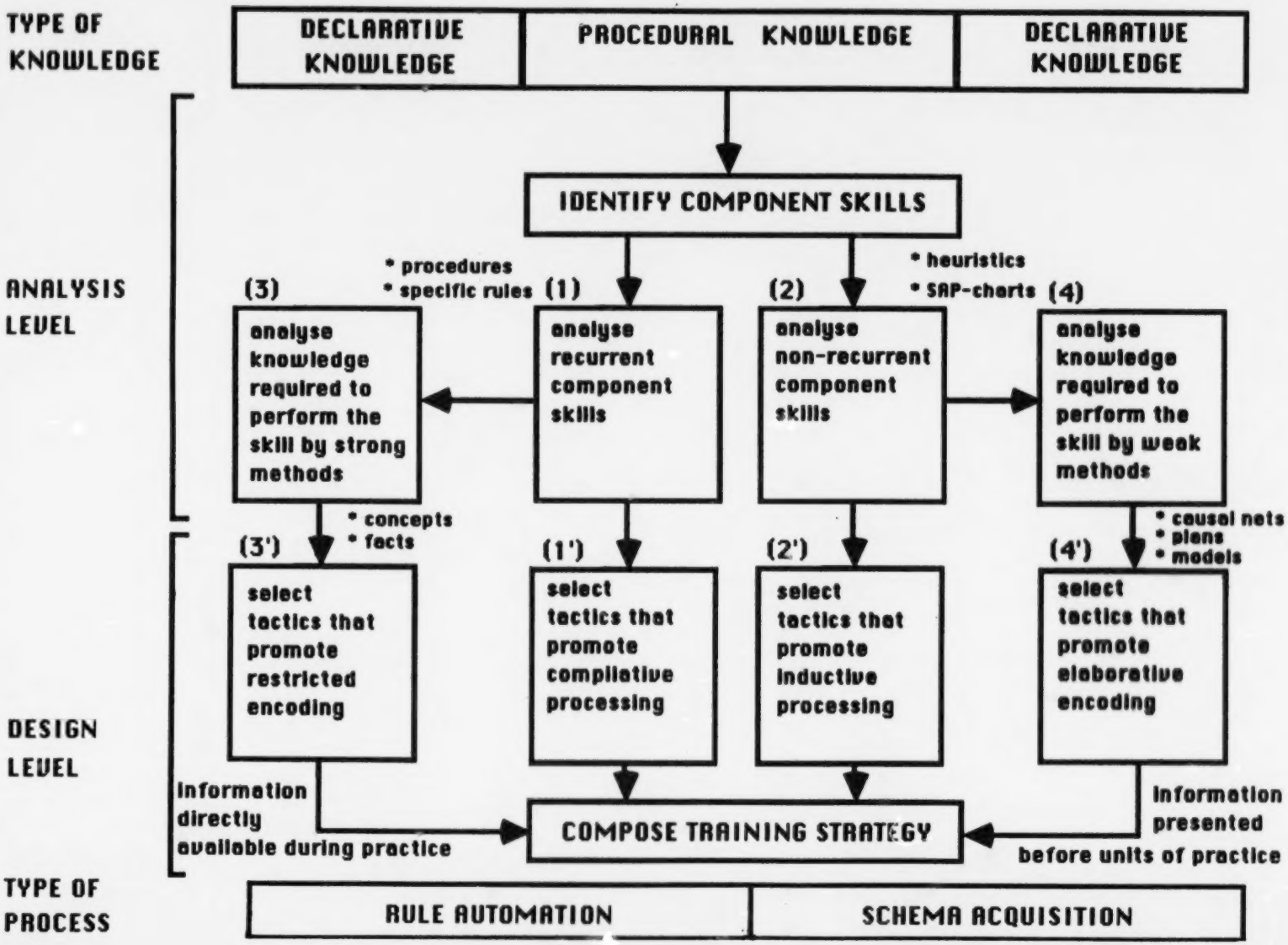
TYPE OF KNOWLEDGE
DECLARATIVE KNOWLEDGE
PROCEDURAL KNOWLEDGE
DECLARATIVE KNOWLEDGE
IDENTIFY COMPONENT SKILLS
ANALYSIS LEVEL
* procedures
* specific rules
(3)
(1)
(2)
* heuristics
* SAP-charts
(4)
analyse knowledge required to perform the skill by strong methods
analyse recurrent component skills
analyse non-recurrent component skills
analyse knowledge required to perform the skill by weak methods
* concepts
* facts
(3')
(1')
(2')
(4')
* causal nets
* plans
* models
select tactics that promote restricted encoding
select tactics that promote compilative processing
select tactics that promote inductive processing
select tactics that promote elaborative encoding
DESIGN LEVEL
information directly available during practice
COMPOSE TRAINING STRATEGY
information presented before units of practice
TYPE OF PROCESS
RULE AUTOMATION
SCHEMA ACQUISITION

FIGURE 1

Fault management

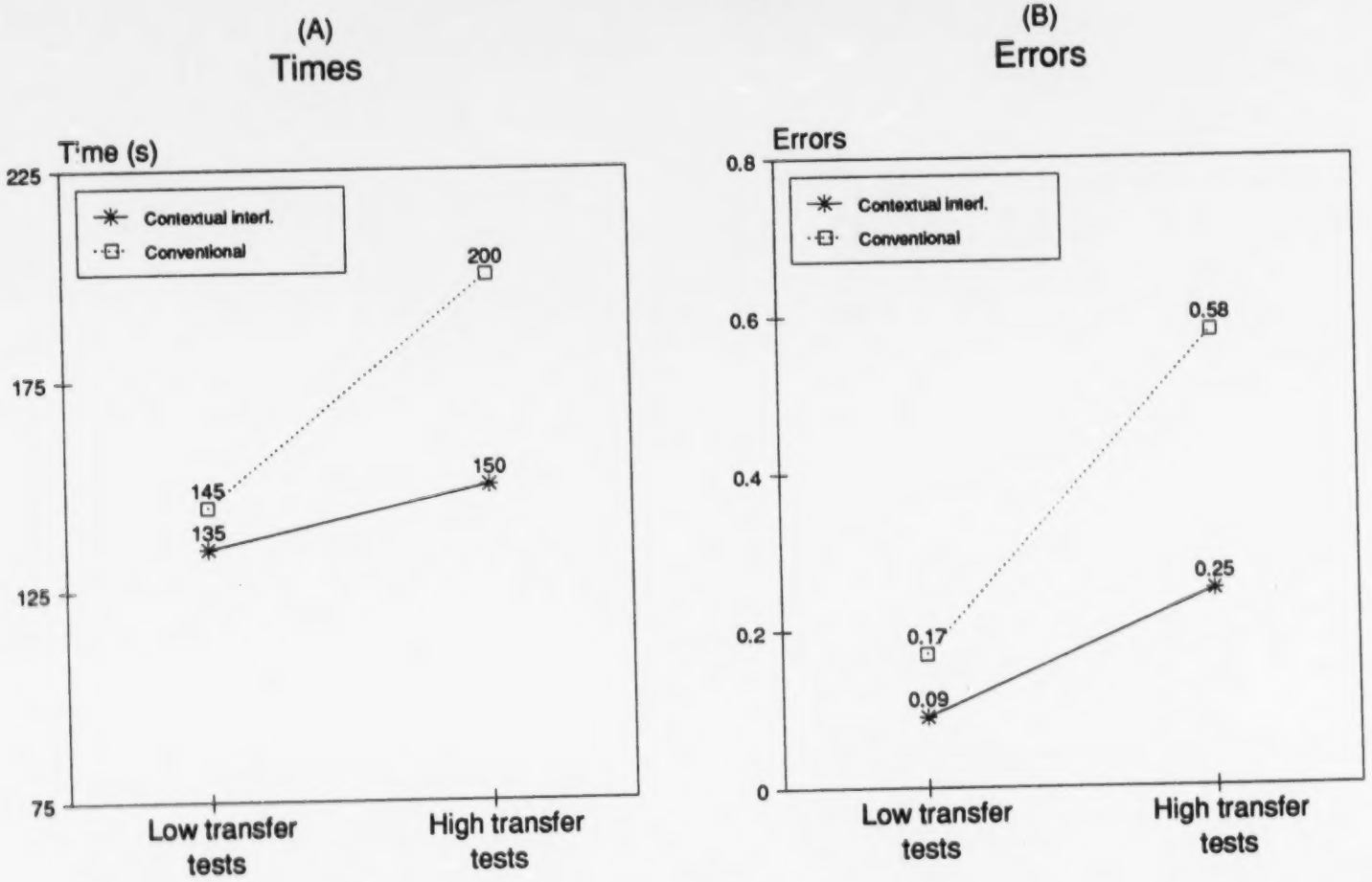

FIGURE 2

Computer programming

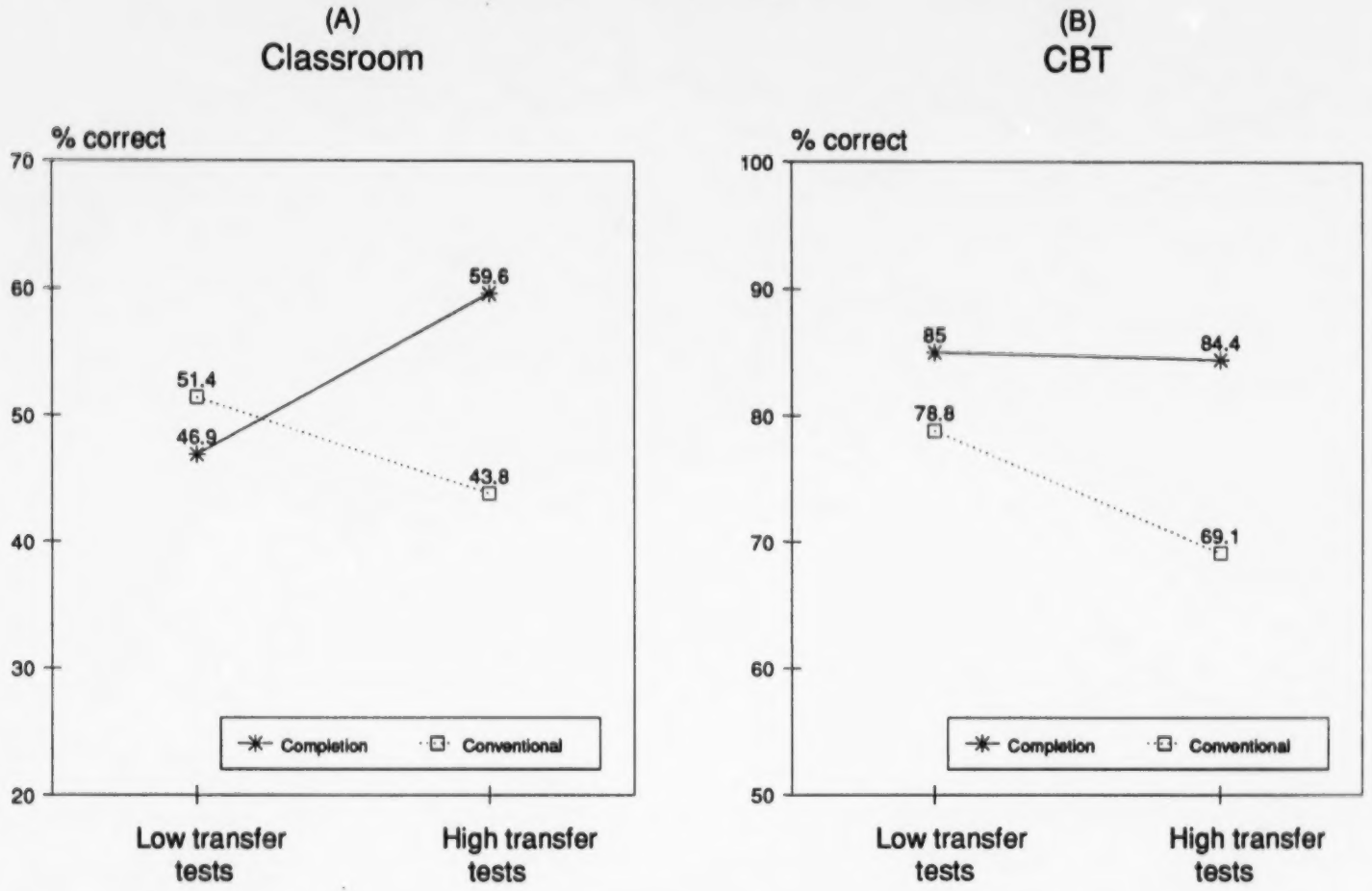

FIGURE 2

Statistical analyzing

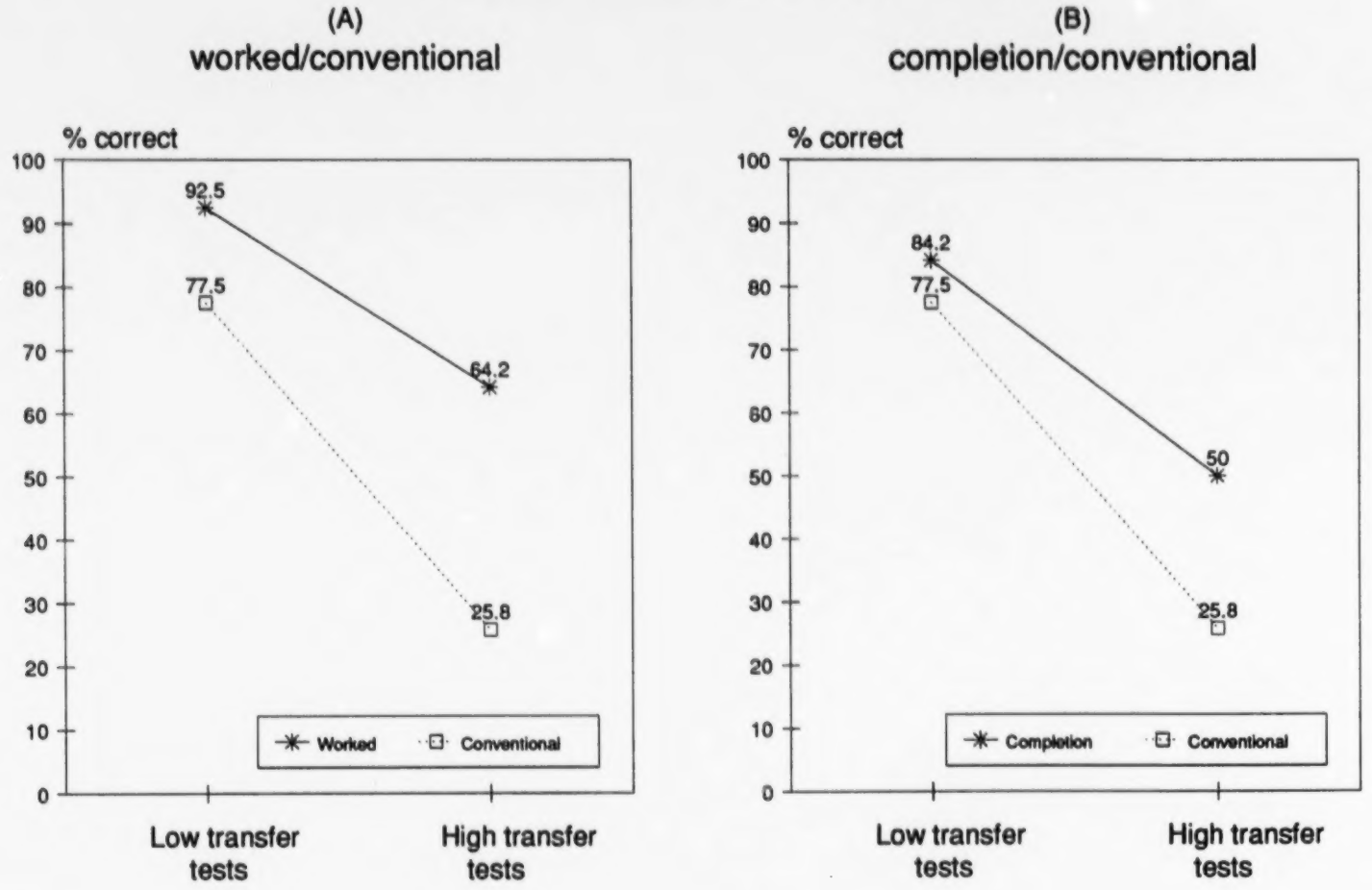

FIGURE 4